ulm university universität ulm

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Konzeption und Realisierung einer Plattform zur Verwaltung von webbasierten Fragebögen

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Fabian Egl
fabian.egl@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Michael Stach

2018

Fassung 10. September 2018

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Fragebögen sind ein wichtiges Mittel der Psychologie um Nutzerbefragungen durchzuführen und somit Forschungsdaten zu erhalten. Zusätzlich zur Psychologie werden Fragebögen auch von Ärzten genutzt um Patientendaten zu erfragen, die momentan nicht auf anderem Weg zu erhalten sind. Jedoch sind diese Fragebögen oft papiergebunden, was eine Verarbeitung dieser erschwert. Für papiergebundene sowie digitale Fragebögen gibt es darüber hinaus keine zentrale Plattform, welche ein Speichern und Verwalten der Fragebögen mit den Antworten ermöglicht. Da somit auch keine zentrale Plattform für einen Austausch der Fragebögen oder der aus diesen gewonnenen Daten existiert, wird der Austausch der Forschungsdaten zwischen Forschern deutlich erschwert.

Ziel dieser Arbeit ist es daher eine webbasierte Plattform zu konzipieren und zu entwickeln, damit Forscher und Ärzte jederzeit über das Internet mit dieser Plattform interagieren können. Die Plattform ermöglicht dabei ein Speichern, Verwalten und Teilen von digitalen Fragebögen. Diese bietet dabei als Schnittstelle eine auf der REST-Architektur basierende API, sowie einen Web-Client an.

Inhaltsverzeichnis

1 Einleitung

In diesem Kapitel wird eine kurze Einführung in die Thematik gegeben, auf der diese Arbeit beruht. Dabei wird der Fokus auf die Motivation und die Problemstellung, welche der Arbeit zu Grunde liegen, gelegt. Der Abschnitt 1.1 besteht aus einer kurzen Hinführung zum Thema mit Einbeziehung der aktuellen Situation, auf der diese Arbeit basiert. Danach folgt der Abschnitt 1.2, welcher die Problemstellung behandelt weshalb diese Arbeit aktuell möglich ist. Abschließend wird im Abschnitt 1.3 eine kurze Zusammenfassung über den Aufbau der Arbeit gegeben. Dabei wird kurz auf den Inhalt aller Kapitel eingegangen.

1.1 Motivation

Momentan gibt es viele Forschungsprojekte, welche mit Hilfe von Fragebögen Nutzerbefragungen, durchführen um Nutzer- / Teilnehmerdaten zu erhalten. Dies wird unter anderem auch in der Psychologie genutzt, um wichtige Forschungsdaten von Patienten zu erhalten und analysieren zu können. Beispiele hierfür sind MobileTx (vergleiche Abschnitt 2.1), TrackYourTinitus (vergleiche Abschnitt **??**) und QuestionSys (vergleiche Abschnitt 2.3). Diese Anwendungen und Systeme sind jedoch oft getrennt voneinander und bieten keine Möglichkeit zu einer einfachen Zusammenarbeit oder einem Austausch von Fragebögen und Forschungsdaten. Es gibt hierfür keine zentrale Plattform welche ein Speichern und Verwalten dieser Fragebögen oder Antworten von Teilnehmern auf diese Fragebögen bietet.

1.2 Problemstellung

Wie im Abschnitt 1.1 motiviert, gibt es momentan keine zentrale Plattform für die Speicherung und Verwaltung von Fragebögen. Da dies auch in der Forschung vorhanden ist, erschwert dies den Austausch von Forschungsergebnissen und Informationen, welche bei der Befragung von Patienten erhoben werden. Eine gemein-

same Plattform könnte eine effizientere Forschung und einen besseren Austausch an Forschungsdaten hervorrufen.

Ziel dieser Arbeit ist es deshalb, eine zentrale Plattform für das Speichern und die Verwaltung von Fragebögen zu erstellen. Diese Plattform sollte dabei webbasiert sein, um die Möglichkeit zu bieten überall verfügbar zu sein. Dies bedeutet unabhängig von der geographischen Position der Nutzer haben diese die Möglichkeit ihre Fragebögen zu erstellen, zu verwalten und mit anderen Nutzern zu teilen. Darüber hinaus können sie auch überall die Antworten auf diese Fragebögen abrufen.

Zusätzlich ist ein zentraler Punkt der Arbeit, dass das System auch eine API beinhaltet. Dies ist wichtig, damit die Nutzer bei Bedarf selbst ein System erstellen können, welches speziell auf Ihre Anforderungen angepasst ist. Das System welches in dieser Arbeit erstellt wird, soll primär die Funktion besitzen die Fragebögen zu speichern und zu verwalten. Zusätzlich wird ein Frontend erstellt, welches die Funktionen zum Verwalten und Speichern der Fragebögen besitzt. Dies hat allerdings nur die Funktion eines Beispiels, wie man das System/die API nutzen kann. Ein einheitliches Frontend ist auch nicht sinnvoll, da verschiedene Nutzergruppen verschiedene Anforderungen an den Aufbau des Frontends besitzen. Somit ist es den Nutzergruppen selbst überlassen, ein Frontend für ihre spezifischen Zwecke zu erstellen und an die API des in dieser Arbeit erstellten Systems anzubinden.

1.3 Aufbau der Arbeit

Diese Arbeit ist in 4 Bereiche aufgeteilt. Den Anfang macht die Hinführung zum Thema, dies wird gefolgt von der Konzeption des Systems. Daraufhin folgt die Implementierung des Systems und zum Abschluss noch eine Zusammenfassung der Arbeit.

Der erste Bereich besteht aus einer Einleitung und Hinführung zum Thema, sowie eine Übersicht über verwandte Arbeiten. Dieser Bereich besteht aus den Kapiteln 1 Einleitung, 2 verwandte Arbeiten und 3 Grundlagen. Das Kapitel 1 Einleitung besteht aus einer allgemeinen Hinführung zum Thema sowie der Problemstellung auf der diese Arbeit basiert. Im Kapitel 2 werden kurz verwandte Arbeiten vorgestellt. Danach folgt Kapitel 3 Grundlagen. Dies beinhaltet eine kurze Erklärung der wichtigsten Begriffe sowie eine kurze Erläuterung zu OAuth und der REST.

Der zweite Bereich besteht aus der Planung und Konzipierung des Systems. Er enthält die Kapitel 4 Anforderungen und 5 Entwurf. In Kapitel 4, wird eine Anfor-

derungsanalyse durchgeführt, um die Anforderungen für das System festzulegen. Das Kapitel 5 beinhaltet einen ersten Entwurf für das System.

Danach folgt der dritte Bereich, welcher sich mit der Implementierung des Systems beschäftigt. Er besteht aus dem Kapitel 6 Implementierung. In diesem Kapitel werden zuerst die für die Implementierung benötigten Technologien beschrieben. Daraufhin werden zwei wichtige Aspekte der Implementierung genauer erläutert und ausführlich behandelt.

Den Abschluss bildet der vierte Bereich mit den Kapiteln 7 Anforderungsabgleich und 8 Zusammenfassung und Ausblick. In Kapitel 7 wird überprüft, ob die Anforderungen welche im Kapitel 4 aufgestellt wurden, erfüllt sind. Danach folgt mit Kapitel 8 das letzte Kapitel der Arbeit, welches eine Zusammenfassung der Arbeit liefert und einen Ausblick auf die Zukunft, basierend auf dieser Arbeit, gibt.

2 Verwandte Arbeiten

In diesem Kapitel wird auf drei verwandte Arbeiten und Forschungsprojekte eingegangen.

2.1 MobileTx

MobileTx [7] [22] [21] ist ein Projekt des Institutes für Datenbanken und Informationssysteme der Uni Ulm. Es beschäftigt sich mit der Unterstützung des therapeutischen Prozesses durch intelligente mobile Geräte, wie Smartphones.

Abbildung 2.1 bietet einen Überblick über die Funktionsweise des Projektes. Patienten bekommen wenn nötig von Therapeuten Aufgaben wie zum Beispiel Hausaufgaben gestellt welcher zur Behandlung genutzt werden. Dabei unterstützen Smartphones und ähnliche Geräte den Patienten beim Ausführen dieser Aufgaben. Diese können dabei die Sensoren des Gerätes nutzen um Daten zu sammeln. Eine weitere Möglichkeit des Sammelns der Patientendaten bieten Fragebögen, die der Patient ausfüllt. Diese werden daraufhin an den Server gesendet. Die dort gespeicherten Daten können von den Therapeuten und den Wissenschaftlern abgerufen werden. Wissenschaftler haben auf diese Weise eine deutlich größere Menge an Forschungsdaten und können somit einfacher die Wirksamkeit der gestellten Aufgaben überprüfen. Zusätzlich kann man durch die Daten neue Erkenntnisse gewinnen und dadurch neue Aufgaben entwickeln.

2.2 TrackYourTinnitus

TrackYourTinnitus [15] ist ein Projekt des Institutes für Datenbanken und Informationssysteme der Uni Ulm. Tinnitus ist dabei ein Wahrnehmen von Ton, obwohl keine physikalische Quelle dafür vorhanden ist. Das Projekt beschäftigt sich mit dem beobachten und analysieren der Tinnituswahrnehmung. Um möglichst viele Daten zu

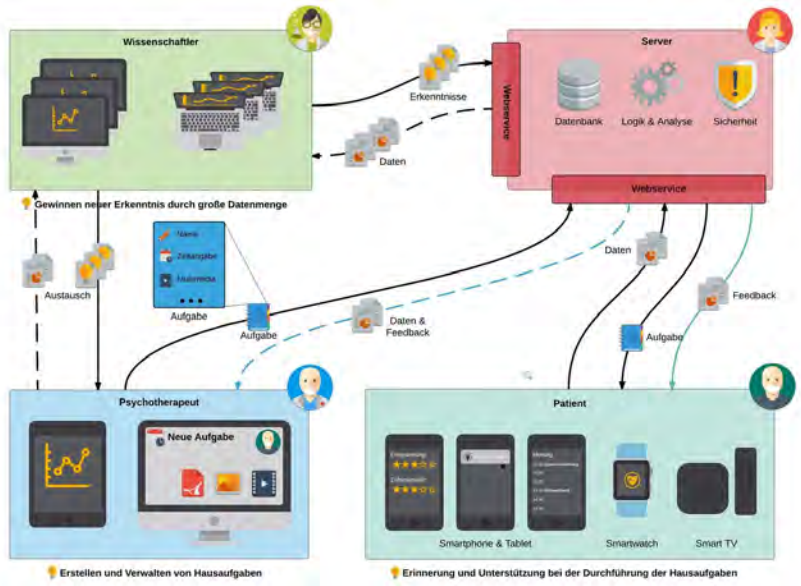

Abbildung 2.1: MobileTx [7] [1]

bekommen nutzt das Projekt crowdsensing. Dabei wird eine Anwendung für Smartphones genutzt, über die die Nutzer entweder zufällig oder auf eigene Initiative Fragebögen zur aktuellen Tinnituswahrnehmung ausfüllen, während das Smartphone gleichzeitig du Umgebungsgeräusche analysiert. Anschließend werden die Daten aus der Nutzerbefragung und der Smartphone Sensoren an die TrackYourTinnitus Datenbank gesendet. Auf dieser Plattform können die gesammelten Daten von Forschern abgerufen und analysiert werden [20].

2.3 QuestionSys

QuestionSys [13] ist ein Projekt des Institutes für Datenbanken und Informationssysteme der Uni Ulm. Das Projekt beschäftigt sich mit dem erstellen eines Fragebogenprozessmodells mit Hilfe dessen einfach Fragebögen digital erstellen, verarbeiten und auswerten zu lassen. Dies hat den Vorteil, dass diese weniger Arbeit benötigen, im Gegensatz zu den momentan hauptsächlich verwendeten papierbasierten Fragebögen in psychologischen Studien.

Abbildung 2.2 zeigt den prozessgetriebenen Ansatz der Architektur des Frameworks, das in diesem Projekt erstellt wurde, um das Sammeln von mobilen Daten zu unterstützen [23].

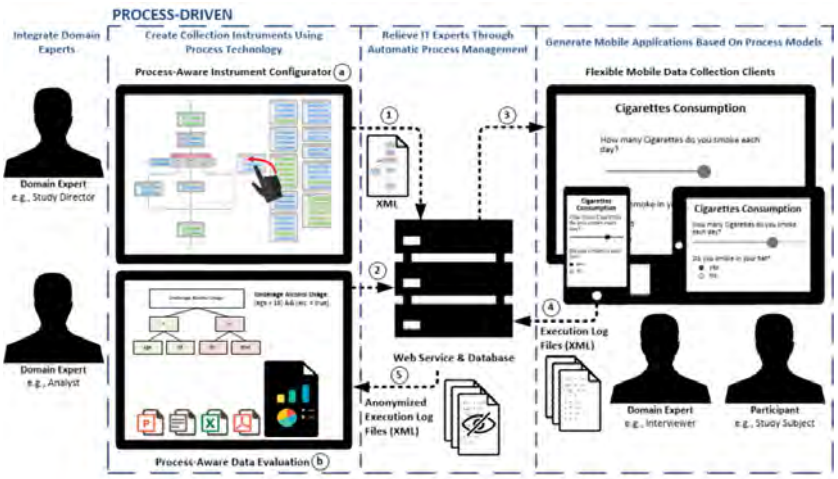

Abbildung 2.2: QuestionSys [13] [2]

3 Grundlagen

Dieses Kapitel beinhaltet die Grundlagen welche für diese Arbeit benötigt werden. Dabei wird in Abschnitt 3.1 eine Übersicht der wichtigsten Begriffe gegeben. Jeder dieser Begriffe wird von einer Erklärung, wie die Begriffe jeweils im Zusammenhang mit dieser Arbeit zu verstehen sind gefolgt. Abschließend folgen eine kurze Zusammenfassung von OAuth in Abschnitt 3.2 und REST in Abschnitt 3.3

3.1 Begrifflichkeiten

Da nicht alle Begriffe die in dieser Arbeit erwähnt werden zu verstehen sind, oder die Art und Weise und der Zusammenhang, wie sie in der Arbeit verwendet werden nicht immer offensichtlich sind, wird in enthält dieser Abschnitt eine Auflistung dieser Begriffe. Danach folgt eine kurze Zusammenfassung oder Erklärung wie diese Begriffe im Zusammenhang mit dieser Arbeit zu verstehen sind. Um eine Suche der Begriffe zu ermöglichen sind diese in alphabetischer Reihenfolge aufgelistet.

API API ist eine Abkürzung für Application Programming Interface. In dieser Arbeit ist eine API eine Schnittstelle zwischen dem erstellten System und Nutzern oder anderen Systemen, welche mit dem erstellten System interagieren möchten. Dabei können die Nutzer oder andere Systeme nur die im System vordefinierten Funktionen für eine Interaktion nutzen.

API-Key Ein API-Key ist ein zufällig generierter String einer vorher festgelegten Länge. Jeder String ist dabei eindeutig und kann somit genau einem Nutzer des Systems zugewiesen werden. Dies wird im System genutzt, um eine eindeutige Authentifizierung des Nutzers beim Nutzen der API zu ermöglichen. Die Erstellung und Verwaltung der API-Keys wird im System durch OAuth [9] umgesetzt. Dieser API-Key entspricht dem Bearer Token im OAuth2.0 Protokol (vergleiche Abschnitt 3.2).

CRUD CRUD steht für Create, Read, Update und Delete [3]. Im Sinne dieser Arbeit wird CRUD genutzt um die Standardaufgaben beim verwalten einer Ressource zusammenzufassen. Es steht hierbei für das Erstellen, Lesen, Editieren und Löschen einer Ressource.

Fragebogen Als Fragebogen werden im Falle dieser Arbeit eine Sammlung von Fragen bezeichnet. Dabei können diese Fragen verschiedene Typen haben, wie zum Beispiel Multiple Choice, freie Antwort, Bewertungen und viele weitere. Für diese Arbeit wird angenommen, dass die Fragebögen auf dem SurveyJS Framework bestehen und als JSON innerhalb der Datenbank abgespeichert werden.

Synonym: Survey

Fragebogenresultat Ein Fragebogenresultat ist eine Sammlung von Antworten für einen gegebenen Fragebogen. Diese Antworten können in einem Fragebogen vorhanden sein oder auch getrennt von diesem gespeichert sein.

Synonym: Survey Result

Register-Token Ein Register-Token besteht wie ein API-Key aus einem zufällig generierten einmaligen String einer festen Länge. Dieser kann von einem Administrator erstellt werden und per E-Mail verschickt werden. Dieser Token wird benötigt um einen neuen Nutzer Account zu registrieren. Somit wird gewährleistet, dass man nur mit Erlaubnis des Administrators einen Account erstellen kann.

Ressource Jede Information die benannt werden kann, kann eine Ressource sein [19].

RESTful API Eine RESTful API ist eine API welche die in Abschnitt 3.3 genannten Bedingungen erfüllt.

Teilnehmer Ein Teilnehmer ist eine Person, welche einen gegebenen Fragebogen beantwortet. Dabei muss der Teilnehmer nicht zwingend selbst die Person sein, welche den Fragebogen ausfüllt. Es besteht auch die Möglichkeit, dass ein Arzt einem Patienten Fragen stellt, sowie die Antworten von diesem dokumentiert und als Fragebogenresultat abspeichert. In diesem Fall ist der Patient der Teilnehmer.

Web-Client Unter dem Web-Client wird in Bezug auf dieses System das komplette Frontend verstanden, das heißt, alle HTML Seiten, die ein Gast, Nutzer oder Administrator abrufen kann (für fast alle Seiten muss der Nutzer bzw. Administrator eingeloggt sein). Diese HTML Seiten beinhalten oft Formulare, mit denen ein Nutzer Ressourcen in der Datenbank verändern kann.

3.2 OAuth

OAuth2.0 [9] ist ein Protokoll, welches eine sichere Autorisierung in einfachen und standardisierten Methoden für Web-, Mobile- und Desktopanwendungen ermöglicht. Die OAuth2.0 Spezifizierung und ihre Erweiterungen werden von der IETF OAuth Working Group entwickelt. Das OAuth Protokoll basiert dabei auf dem in 2006 erstellten OAuth Protokoll.

Basierend auf dem OAuth Protokoll können einfach APIs erstellt werden. Dabei benötigt man einen Besitzer von Ressourcen. Diese Ressourcen sind auf einem Server gespeichert welcher von OAuth geschützt werden. Darüber hinaus gibt es noch einen Client. Mit diesem Client kann der Besitzer der Ressource nach erfolgreicher Authentifizierung die Ressourcen vom Server abrufen. Damit er diese Abrufen kann benötigt er Access Tokens. Diese Access Tokens können von einem Autorisierungsserver erstellt werden, nachdem dieser die Einwilligung des Nutzer, dem die Ressourcen gehören erhalten hat. Das Ursprüngliche OAuth Protokoll benötigt bei jeder Anfrage eine kryptographische Signatur des Ressourcen Besitzers, damit die Identität und Autorisierung des Clients bestätigt werden kann. In OAuth2.0 wurden diese kryptographische Signaturen durch Bearer Tokens ersetzt. Bearer Tokens sind einfache Access Tokens bei denen der Besitzt dieser Ausreicht, um auf die API zugreifen zu können. Für API Zugriffe werden keine weiteren Informationen benötigt. [18]

3.3 REST

Representational State Transfer (REST) [19] ist ein Architekturstil für distributed hypermedia systems. Damit ein System RESTful ist, müssen die nachfolgenden sechs Bedingungen erfüllt sein.

Die erste Bedingung ist eine Client-Server Architektur. Die zweite Bedingung ist die Zustandslosigkeit der Kommunikation zwischen Client und Server. Das heißt auf

dem Server werden keine Daten bezüglich der Session gespeichert, der Server verarbeitet jeweils nur die pro Anfrage eingegangenen Daten. Die dritte Bedingung ist eine Möglichkeit Daten auf Client-Seite im Cache abzuspeichern, sofern diese im Cache abgespeichert werden dürfen. Die vierte Bedingung ist ein einheitliches Interface. Die fünfte Bedingung ist ein schichtbasiertes System. Dabei haben die einzelnen Schichten verschiedene Hierarchien. Dies kann genutzt werden um nur bestimmte Funktionen des Systems in bestimmten Ebenen sichtbar und somit verfügbar zu machen. Die letzte Bedingung ist Code-On-Demand. Dies erlaubt es dem Client Applets oder Skripte zu downloaden und auszuführen.

4 Anforderungsanalyse

Bevor das System Implementiert werden kann, muss zuerst klar sein welche Anforderungen das System erfüllen soll. Deshalb wird in diesem Kapitel eine Anforderungsanalyse des Systems ausgeführt.

Dieses Kapitel ist in zwei Abschnitte der Anforderungsanalyse aufgeteilt. Zuerst erfolgt in Abschnitt 4.1 die Analyse des Systems. Daraufhin werden im Abschnitt 4.2 die Anforderungen des Systems erstellt.

4.1 Analyse

In diesem Abschnitt erfolgt die Analyse des Systems. Hierfür wird zuerst ein Szenario aufgestellt, um herauszufinden wie das System später genutzt wird. Daraufhin folgt die Benutzerprofilanalyse. In dieser werden alle Gruppen von Benutzern aufgeführt, welche das System später nutzen wollen. Diese Gruppen werden daraufhin in Bezug auf die Systemnutzung analysiert. Abschließend werden noch alle Randbedingungen aufgeführt und kurz erklärt.

4.1.1 Szenario

Bevor die Anforderungen des Systems aufgestellt werden können sollten zuerst die wichtigsten Szenarien der Systemnutzung aufgestellt werden, damit eindeutig definiert ist, wie das System später genutzt wird. In diesem Abschnitt werden die wichtigsten Szenarien der Systemnutzung aufgestellt. Dies wird am Anfang der Anforderungsanalyse erstellt, damit in Abschnitt 4.2 die Anforderungen des Systems basierend auf diesem Szenarien aufgestellt werden.

Wie in Abschnitt 1.1 und Abschnitt 1.2 bereits genannt, ist ein wichtiges Benutzerszenario für diese Arbeit das Speichern und Verwalten von Fragebögen in der Forschung (vor allem in der Psychologie). Dabei soll die Plattform eine zentrale

Plattform sein, welche für alle beteiligten Personen immer und möglichst überall erreichbar sein sollte.

4.1.2 Benutzerprofilanalyse

Damit das System an die Benutzer angepasst und nicht an den Benutzern vorbei entwickelt wird, wird in diesem Abschnitt eine Benutzerprofilanalyse ausgeführt. Bei dieser wird analysiert, welche Nutzergruppen das fertige System benutzen werden. Darüber hinaus wird aufgeführt, wie die einzelnen Nutzergruppen mit dem System interagieren.

Das System wird wie in Abschnitt 1.1 erwähnt für Forschungsprojekte entwickelt. Somit bilden die Forscher und Ärzte eine wichtige Nutzergruppe. Die Forscher werden im späteren System Fragebögen erstellen, verwalten und eventuell mit ähnlichen Forschungsprojekten teilen. Darüber hinaus ist für Forscher die Analyse der Antworten auf die Fragebögen sehr wichtig, weshalb das Abrufen der Antworten auf die im System gespeicherten Fragebögen eine zentrale Funktion für die Forscher ist.

Eine weitere Nutzergruppe bilden die befragten Nutzer und Patienten. Die Nutzer dieser Gruppe benötigen als Funktionen im späteren System im Grunde nur das Abrufen der Fragebögen die sie beantworten sollen und das Erstellen und Abspeichern der Antworten auf diese.

4.1.3 Randbedingungen

Basierend auf Abschnitt 4.1.1 und Abschnitt 4.1.2 können nun die Randbedingungen aufgestellt werden.

Das zu erstellende System soll webbasiert sein und eine API besitzen. Die API ist dabei eine RESTful-API (vergleiche Abschnitt 3.3). Die Autorisierung der API erfolgt mit OAuth (vergleiche Abschnitt 3.2). Als Basis für das Fragebogenschema wird SurveyJS [14] (vergleiche Abschnitt **??**) genutzt.

4.2 Anforderungen

Nachdem nun die grundlegende Analyse des Systems fertig ist, werden nun die darauf aufbauenden Anforderungen des Systems erstellt. In diesem Abschnitt wird

dafür zuerst ein Anwendungsfalldiagramm aufgestellt, welches eine Übersicht über die Anwendungsfälle gibt, welche das System erfüllen soll. Daraufhin folgt eine Zusammenstellung der funktionalen Anforderungen und nicht funktionalen Anforderungen. Diese werden dabei wiederum jeweils von Erklärungen der einzelnen Anforderungen gefolgt.

4.2.1 Anwendungsfalldiagramm

Um das System erfolgreich erstellen zu können, müssen im Vorfeld alle Anwendungsfälle, die das spätere System ausführen können soll, erarbeitet werden. Um diese grafisch darzustellen wurde ein UML-Anwendungsfalldiagramm genutzt. Dieses bietet eine Übersicht über die Nutzer des Systems und die Anwendungsfälle welche diese jeweils am fertigen System ausführen.

In Abbildung 4.1 ist das Anwendungsfalldiagramm für diese Arbeit zu sehen. Da dieses Diagramm sehr viele Anwendungsfälle enthält, wird nachfolgend nur eine kurze Übersicht über die Rollen und die dazugehörigen Anwendungsfälle gegeben. Eine genaue Erklärung der einzelnen Anwendungsfälle befindet sich in Abschnitt 4.2.2, da diese fast identisch zu den funktionalen Anforderungen des Systems sind.

Zur Erklärung des Diagramms folgt eine kurze Übersicht der Rollen mit zugehörigen Anwendungsfällen. Eine genauere Erklärung der einzelnen Rollen folgt in Abschnitt 5.3.

Das System besteht aus drei Rollen. Diese sind der Gast, der Administrator und der Nutzer. Der Gast kann sich im System registrieren und anmelden. Der Administrator kann Nutzer des Systems verwalten. Die Rolle Nutzer bezeichnet den allgemeinen Nutzer des Systems, er kann sich dabei API-Keys erstellen und verwalten um auf die API zugreifen zu können. Darüber hinaus kann er Projekte erstellen und verwalten. Die Berechtigungen hierfür sind für jeden Nutzer und jedes Projekt individuell. Dabei kann ein Nutzer entweder nicht zu einem Projekt gehören, oder er ist Projektverwalter oder Projektteilnehmer in diesem Projekt. Im Diagramm 4.1 kann dabei abgelesen werden, welche Funktionen er in welcher Rolle je Projekt ausführen kann. Eine genauere Erklärung der Rollen mit Berechtigungen der Rolle innerhalb eines Projektes befindet sich wie bereits vorher genannt in Abschnitt 5.3.

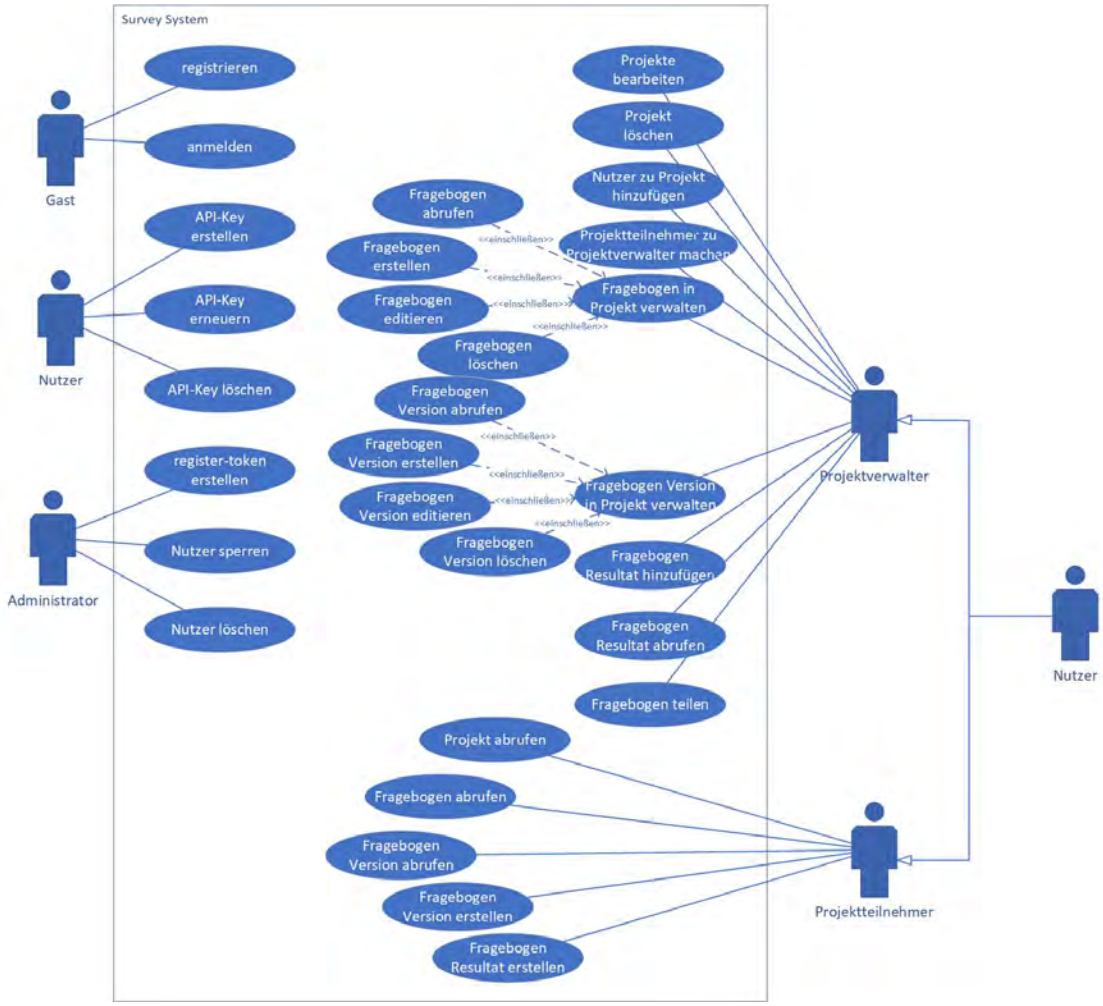

Abbildung 4.1: Anwendungsfalldiagramm

Nr.	Funktionale Anforderung	Seite
Gast:		
FA1.01	Gast kann sich registrieren	17
FA1.02	Gast kann sich anmelden	18
Nutzer:		
FA1.03	Nutzer kann API-Key erstellen	18
FA1.04	Nutzer kann API-Key erneuern	18
FA1.05	Nutzer kann API-Key löschen	18
FA1.06	Nutzer kann neues Projekt erstellen	18
Administrator:		
FA1.07	Administrator kann neuem Nutzer Register-Token schicken	18
FA1.08	Administrator kann Nutzer sperren	19
FA1.09	Administrator kann Nutzer löschen	19

Tabelle 4.1: Nutzerabhängige funktionale Anforderungen

4.2.2 Funktionale Anforderungen

Die funktionalen Anforderungen des Projektes sind zur Übersichtlichkeit in drei Tabellen aufgeteilt. Die Tabelle 4.1 enthält alle allgemeinen Nutzeranforderungen. Dies wird von der Tabelle 4.2 gefolgt welche die speziellen rollenspezifischen Anforderungen innerhalb eines Projektes beinhalten. Die Tabelle 4.3 beinhaltet alle nutzerunabhängigen Anforderungen.

Eine genaue Erklärung der einzelnen Anforderungen befindet sich im Anschluss an die Tabellen. Alle Anforderungen die einen Nutzer des Systems betreffen, der kein Gast ist, setzen voraus, dass der Nutzer im System entweder durch Anmeldung im Web-Client oder durch Angabe des API-Keys beim Nutzen der API autorisiert ist. Bei Funktionen, welche ein Projekt betreffen, wird bei den Anforderungen vorausgesetzt, dass der Nutzer die entsprechenden Berechtigungen innerhalb dieses Projektes hat (entspricht Rolle Projektverwalter oder Projektteilnehmer innerhalb des Projekts).

FA1.01 Gast kann sich registrieren

Ein Gast kann ein HTML-Formular abrufen mit dem er sich anschließend in der Anwendung registrieren kann. Um dieses Formular erfolgreich nutzen zu können wird ein Register-Token benötigt. Ein Administrator kann diesen Token erstellen und dem neuen Nutzer per E-Mail zuschicken.

FA1.02 Gast kann sich anmelden

Ein Gast kann ein HTML-Formular zum anmelden im System abrufen und sich mit Hilfe von diesem im System anmelden. Der Web-Client des Systems kann nur nach erfolgreicher Anmeldung eines Nutzers genutzt werden.

FA1.03 Nutzer kann API-Key erstellen

Ein Nutzer kann über die API einen neuen API-Key erstellen.

Alternativ kann ein Nutzer auch ein HTML-Formular abrufen mit dem er sich einen API-Key erstellen kann. Mit diesem API-Key kann er auf die API zugreifen.

FA1.04 Nutzer kann API-Key erneuern

API-Keys können standardmäßig nur für eine Begrenzte Zeit genutzt werden. Ein Nutzer kann über die API seinen API-Key erneuern. Dabei wird die Gültigkeit des API-Keys auf die Standarddauer des Systems vom aktuellen Zeitpunkt aus gesetzt.

Alternativ kann ein Nutzer auch ein HTML-Formular zum erneuern seines API-Keys abrufen und damit seinen API-Key erneuern.

FA1.05 Nutzer kann API-Key löschen

Ein Nutzer kann seinen API-Key über die API löschen.

Alternativ kann ein Nutzer ein HTML-Formular abrufen mit dem er seinen API-Key löschen kann.

FA1.06 Nutzer kann neues Projekt erstellen

Ein Nutzer kann ein neues Projekt über die API erstellen. Dabei muss er dem System alle benötigten Ressourcen übermitteln, welche zum erstellen des Projektes benötigt werden.

Alternativ kann ein Nutzer auch ein HTML-Formular abrufen mit dem er ein neues Projekt erstellen kann.

FA1.07 Administrator kann neuem Nutzer Register-Token schicken

Um einen neuen Nutzer Account erstellen zu können wird ein Register-Token benötigt. Um diesen Token zu erstellen, kann der Administrator ein HTML-Formular

abrufen. Dieses Formular kann vom Administrator genutzt werden um einen neuen Register-Token zu erstellen und dem neuen Nutzer an dessen E-Mail zu senden. Dabei muss der Administrator die E-Mail des neuen Nutzers kennen.

FA1.08 Administrator kann Nutzer sperren

Ein Administrator kann ein HTML-Formular abrufen um einen Nutzer bei gegebener E-Mail zu sperren. Der gesperrte Nutzer kann daraufhin das System nicht mehr nutzen, während er gesperrt ist. Dies schließt die Nutzung des Web-Clients sowie der API des Systems ein.

FA1.09 Administrator kann Nutzer löschen

Ein Administrator kann ein HTML-Formular abrufen um einen Nutzer bei gegebener E-Mail zu löschen.

FA2.01 Projektverwalter kann Projekte bearbeiten

Ein Nutzer kann ein existierendes Projekt, bei dem er Projektverwalter ist, über die API editieren. Dabei übergibt er dem System das Projekt und die neuen Werte, welche das Projekt annehmen soll.

Alternativ kann der Nutzer ein HTML-Formular zum editieren des Projektes abrufen. Mit diesem Formular kann der Nutzer die Daten des Projektes editieren und danach abspeichern.

FA2.02 Projektverwalter kann Projekte löschen

Ein Nutzer kann ein existierendes Projekt, bei dem er Projektverwalter ist, über die API löschen.

Alternativ kann der Nutzer ein HTML-Formular zum löschen des Projektes abrufen. Mit diesem Formular kann der Nutzer das Projekt löschen.

FA2.03 Projektverwalter kann Nutzer als Projektteilnehmer zu Projekt hinzufügen

Ein Nutzer kann zu einem existierenden Projekt, bei dem er Projektverwalter ist, andere Nutzer über die API als Projektteilnehmer hinzufügen.

Nr.	Funktionale Anforderung	Seite
Projektverwalter:		
FA2.01	Projektverwalter kann Projekte bearbeiten	19
FA2.02	Projektverwalter kann Projekte löschen	19
FA2.03	Projektverwalter kann Nutzer als Projektteilnehmer zu Projekt hinzufügen	19
FA2.04	Projektverwalter kann Projektteilnehmer zu Projektverwalter des Projektes machen	21
FA2.05	Projektverwalter kann CRUD Fragebogen innerhalb des Projektes	21
FA2.06	Projektverwalter kann CRUD Fragebogenversion innerhalb einem Fragebogen des Projektes	21
FA2.07	Projektverwalter kann Fragebogenresultat zu einem Fragebogen des Projektes hinzufügen	21
FA2.08	Projektverwalter kann Fragebogenresultat-Kollektion zu einem Fragebogen des Projektes abrufen	22
FA2.09	Projektverwalter kann Fragebogen-Schema mit anderen Nutzern teilen	22
Projektteilnehmer:		
FA2.10	Projektteilnehmer kann Projekte abrufen	22
FA2.11	Projektteilnehmer kann Fragebögen des Projektes abrufen	22
FA2.12	Projektteilnehmer kann Versionen eines Fragebogens des Projektes abrufen	23
FA2.13	Projektteilnehmer kann Version eines Fragebogens des Projektes erstellen	23
FA2.14	Projektteilnehmer kann Resultat eines Fragebogens des Projektes erstellen	23

Tabelle 4.2: Projektspezifische funktionale Anforderungen

Alternativ kann der Nutzer ein HTML-Formular abrufen mit dem er andere Nutzer als Projektteilnehmer zu diesem Projekt hinzufügen kann.

FA2.04 Projektverwalter kann Projektteilnehmer zu Projektverwalter des Projektes machen

Ein Nutzer kann Projektteilnehmer eines Projektes, bei dem er Projektverwalter ist, zu Projektverwalter des Projektes über die API machen.

Alternativ kann der Nutzer ein HTML-Formular abrufen um Projektteilnehmer des Projektes zu Projektverwaltern zu machen. Mit diesem HTML-Formular kann er Projektteilnehmer des Projektes, zu Projektverwaltern machen.

FA2.05 Projektverwalter kann CRUD Fragebogen innerhalb des Projektes

Ein Nutzer der Projektverwalter eines Projektes ist, kann alle CRUD Funktionen auf die Fragebogen-Ressource über die API anwenden, wenn die Ressource zu dem Projekt zugehörig ist. Das heißt der Nutzer kann einen Fragebogen über die API erstellen, lesen, editieren und löschen.

Alternativ kann der Nutzer ein HTML-Formular zum erstellen, editieren und löschen einer Fragebogen-Ressource des Projektes abrufen und ausführen. Zusätzlich kann er über den Web-Client auch die Fragebogen-Ressource abrufen.

FA2.06 Projektverwalter kann CRUD Fragebogen-Version innerhalb einem Fragebogen des Projektes

Ein Nutzer der Projektverwalter eines Projektes ist, kann zu allen zugehörigen Fragebogen-Ressourcen des Projektes, die CRUD Funktionen auf der Fragebogenversion-Ressource ausführen.

Alternativ kann der Nutzer zu allen zugehörigen Fragebogen Ressourcen des Projektes, HTML-Formulare zum erstellen, editieren und löschen der Fragebogenversion-Ressource abrufen und ausführen. Darüber hinaus kann er über den Web-Client die Fragebogenversion-Ressource auch abrufen.

FA2.07 Projektverwalter kann Fragebogenresultat zu einem Fragebogen des Projektes hinzufügen

Ein Nutzer, der Projektverwalter eines Projektes ist, kann zu allen zugehörigen Fragebogen-Ressourcen eine Fragebogenresultat-Ressource erstellen und über die API hinzufügen.

Alternativ kann der Nutzer auch ein HTML-Formular zum erstellen einer Fragebogenresultat-Ressource erstellen und zu einer Fragebogen-Ressource des Projektes über das HTML-Formular hinzufügen.

FA2.08 Projektverwalter kann Fragebogenresultat-Kollektion zu einem Fragebogen eines Projektes abrufen.

Ein Nutzer kann zu einem Projekt, bei dem er Projektverwalter ist, zu allen Fragebogen-Ressourcen des Projektes eine Sammlung aller Fragebogenresultat-Ressourcen zu der gewählten Fragebogen-Ressource über die API abrufen.

Alternativ kann der Nutzer die Sammlung aller Fragebogenresultat-Ressourcen einer Fragebogen-Ressource des Projektes über den Web-Client abrufen.

FA2.09 Projektverwalter kann Fragebogen-Schema mit anderen Nutzern teilen

Ein Nutzer der Projektverwalter eines Projektes ist, kann eine Fragebogenressource des Projektes über die API mit anderen Nutzern teilen. Dabei müssen die beiden Nutzer nicht im selben Projekt sein.

Alternativ kann der Nutzer auch ein HTML-Formular abrufen und über den Web-Client mit dem er eine Fragebogenressource mit anderen Nutzern teilen kann.

FA2.10 Projektteilnehmer kann Projekte abrufen

Ein Nutzer, der Projektteilnehmer eines Projektes ist, kann die Projekt-Ressource über die API abrufen.

Alternativ kann er ein HTML-Formular abrufen und über den Web-Client ausführen, mit dem er eine Projekt-Ressource abrufen kann.

FA2.11 Projektteilnehmer kann Fragebögen des Projektes abrufen

Ein Nutzer der Projektteilnehmer eines Projektes ist, kann über die API die Fragebogen-Ressourcen des Projektes einzeln oder als Sammlung aller abrufen.

Alternativ kann der Nutzer dies auch über den Web-Client abrufen.

Nr.	Funktionale Anforderung	Seite
Fragebögen:		
FA3.01	Fragebogen hat Versionsstände	23
FA3.02	Fragebogen wird validiert	24
FA3.03	Fragebogen-Schema kann über HTML abgerufen werden	24
FA3.04	Fragebogen-Schema kann als JSON abgeurfen werden	24

Tabelle 4.3: Allgemeine funktionale Anforderungen

FA2.12 Projektteilnehmer kann Versionen eines Fragebogens des Projektes abrufen

Ein Nutzer, der Projektteilnehmer eines Projektes ist, kann über die API Fragebogenversion-Ressourcen einzeln oder als Sammlung aller Ressourcen abrufen.

Alternativ kann der Nutzer dies auch über den Web-Client abrufen.

FA2.13 Projektteilnehmer kann Version eines Fragebogens des Projektes erstellen

Ein Nutzer, der Projektteilnehmer eines Projektes ist, kann eine neue Fragebogenversion-Ressource erstellen und diese über die API abspeichern.

Alternativ kann der Nutzer ein HTML-Formular zum erstellen einer neuen Fragebogenversion-Ressource abrufen. Mit diesem Formular kann er die erstellte Ressource über den Web-Client abspeichern.

FA2.14 Projektteilnehmer kann Resultat eines Fragebogens des Projektes erstellen

Ein Nutzer, der Projektteilnehmer eines Projektes ist, kann eine neue Fragebogenresultat-Ressource erstellen und diese über die API abspeichern.

Alternativ kann der Nutzer ein HTML-Formular zum erstellen einer neuen Fragebogenresultat-Ressource abrufen. Mit diesem Formular kann er die erstellte Ressource über den Web-Client abspeichern.

FA3.01 Fragebogen hat Versionsstände

Jede Fragebogen-Ressource kann verschiedene Versionen haben.

FA3.02 Fragebogen wird validiert

Eine Fragebogen-Ressource kann zumindest clientseitig validiert werden.

FA3.03 Fragebogen-Schema kann über HTML abgerufen werden

Das Fragebogen-Schema einer Fragebogen Ressource kann mindestens mit dem Web-Client als HTML abgerufen werden. Dabei wird das Schema der Ressource direkt als fertige Survey mit Hilfe des SurveyJS Framework angezeigt.

FA3.04 Fragebogen-Schema kann als JSON abgerufen werden

Das Fragebogen Schema einer Fragebogen-Ressource kann mindestens über die API im JSON Format abgerufen werden. Dabei wird die JSON in einem Format übergeben, welches von SurveyJS als Survey gerendert werden kann.

4.2.3 Nicht funktionale Anforderungen

Nicht funktionale Anforderungen sind alle Anforderungen, welche sich nicht auf die Funktionen des Systems beziehen.

Nr.	Nicht funktionale Anforderung	Seite
NFA1	Datenschutz	25
NFA2	Datenminimierung	25
NFA3	Verfügbarkeit	25
NFA4	Datenintegrität	25
NFA5	Skalierbarkeit	25
NFA6	Robustheit	25

Tabelle 4.4: Nicht funktionale Anforderungen

NFA1 Datenschutz

Die Daten im System dürfen nur durch befugte Nutzer abgerufen oder verändert werden. Darüber hinaus sollen die Daten wenn möglich Anonym gespeichert werden.

NFA2 Datenminimierung

Im System sollen nur die personenbezogenen Daten gespeichert werden, welche unbedingt für das System nötig sind.

NFA3 Verfügbarkeit

Das System soll für befugte Nutzer jederzeit von überall über das Internet erreichbar und funktionsfähig sein.

NFA4 Datenintegrität

Die im System gespeicherten Daten dürfen nur durch befugte Nutzer verändert werden.

NFA5 Skalierbarkeit

Das System soll so erstellt werden, das eine Erweiterung um zusätzliche Funktionen nicht umständlich ist.

NFA6 Robustheit

Fehler im System sollten sinnvoll abgefangen werden und nicht zu einem kompletten Systemausfall oder Ausfall eines Teilsystems führen.

5 Entwurf

Im Kapitel Entwurf wird der grundlegende Entwurf der Architektur und der Nutzer in Verbindung mit den entsprechenden Rollen aufgestellt. Dabei basiert der Entwurf auf den Anforderungen, welche im vorherigen Kapitel 4 aufgestellt wurden.

5.1 Architektur

Um die Funktion eines Web-Frontends und einer API den Nutzern zur Verfügung stellen zu können, benötigt das System mindestens drei Komponenten die miteinander interagieren. Das wichtigste Teilsystem ist dabei das Backend, dort werden die meisten Funktionen des späteren Systems ausgeführt. Darüber hinaus benötigt das System noch zwei Schnittstellen mit denen die späteren Nutzer mit dem System interagieren können. Dies wird zum einen durch den Web-Client und zum anderen von dem Web-Service (API) übernommen. Siehe Abbildung 5.1.

5.1.1 Web-Client

Der Web-Client ist eine Schnittstelle zwischen dem System und den Nutzern, welche ein HTML Frontend anbietet. Die Nutzer können hier HTML-Seiten abrufen, um Daten vom Backend des Systems angezeigt zu bekommen. Außerdem können auch HTML-Seiten abgerufen werden, welche eine HTML-Formular besitzen, mit dem Nutzer Daten an das Backend des Systems schicken können damit diese dort verarbeitet werden können. Die Interaktion von Nutzern mit dem System findet hierbei mit Hilfe von HTTP-Anfrage (entspricht HTTP-Request) und HTTP Antwort (entspricht HTTP-Response) statt. Der HTTP-Anfrage wird hier von den Nutzern bzw. deren Browsern an das System geschickt, diese bekommen vom System ein HTTP-Antwort als Antwort. Der HTTP-Anfrage besteht aus HTML welches vom Browser angezeigt wird und kann auch zusätzliche Daten wie zum Beispiel Fragebögen enthalten.

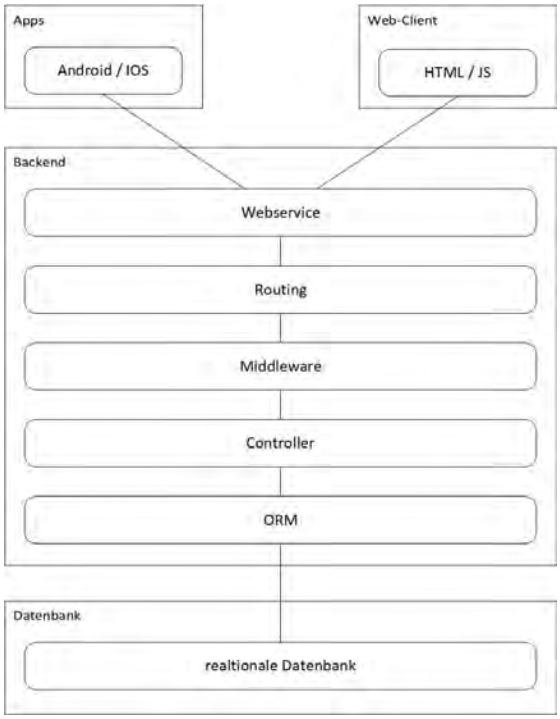

Abbildung 5.1: Architektur

Um den Web-Client nutzen zu können, muss ein Nutzer im System eingeloggt sein.

5.1.2 Web-Service

Der Web-Service funktioniert ähnlich wie der Web-Client. Jedoch findet hier die Kommunikation zwischen Nutzer und System durch JSON-Anfrage (entspricht JSON-Request) und JSON-Antwort (entspricht JSON-Response) statt. Der Nutzer sendet auch hier ein Anfrage an das System, welcher in diesem Fall jedoch aus einer JSON besteht. Als Antwort bekommt der Nutzer ein JSON-Antwort enthält. Die JSON-Antwort enthält dabei ein Statuscode welcher dem Nutzer mitteilt, ob die vom Nutzer im Anfrage gewählte Funktion erfolgreich ausgeführt wurde. Zusätzlich kann die Antwort bei erfolgreicher Ausführung des Anfrage auch weitere Ressourcen im JSON enthalten. Eine genaue Erklärung zum Web-Service mit allen Statuscodes befindet sich im Abschnitt 6.2.1 .

Um den Web-Service nutzen zu können muss sich der Nutzer mit Hilfe des OAuth Passport im JSON-Anfrage gegenüber dem System authentifizieren.

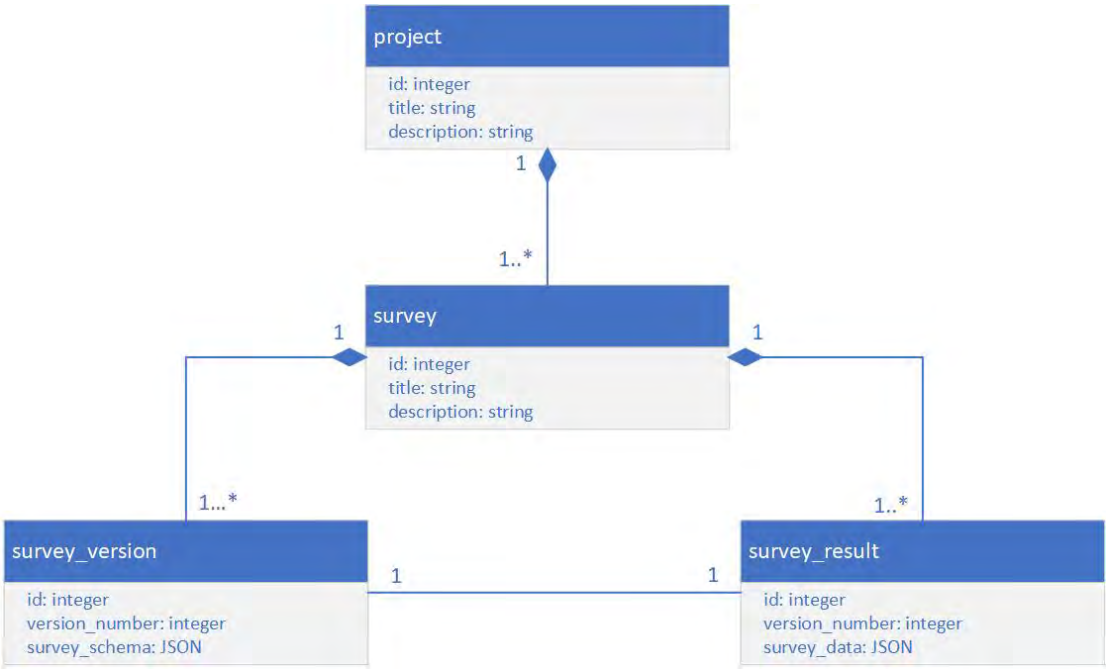

Abbildung 5.2: Entwurf Klassendiagramm

5.1.3 Backend

Das Backend beinhaltet die eigentliche Funktionalität des Systems. Hier werden alle von Nutzern eingehenden Anfragen bearbeitet. Anschließend antwortet das System dem Nutzer mit einer entsprechenden Antwort. Eine Bearbeitung von Anfragen bezeichnet in diesem Fall meist das Erstellen, Lesen, Speichern, Editieren und Löschen von Datensätzen innerhalb des Systems.

5.2 Aufbau der Ressourcen

Im System sollen Fragebögen gespeichert und verwaltet werden. Eine einzelne Speicherung aller Fragebögen macht wenig Sinn. Deshalb bietet das System Projekte an. Fragebögen können nur innerhalb eines Projektes gespeichert werden. Durch Projekte können die Fragebogen zum Beispiel Thematisch gruppiert werden. Ein Fragebogen kann verschiedene Versionen und Resultate enthalten. Eine genauere Darstellung findet sich im folgenden Klassendiagramm (Abbildung 5.2).

5.2.1 Projekt

Ein Projekt ist ein Objekt welches einen Titel und eine Beschreibung sowie eine Sammlung beliebig vieler Fragebögen enthält. Der Titel und die Beschreibung werden dabei genutzt um eine Identifizierung des Projektes und des Inhalts des Projektes einfacher zu machen.

5.2.2 Fragebogen

Ein Fragebogen besitzt genau wie ein Projekt einen Titel und eine Beschreibung. Auch hier wird dies genutzt um eine Identifizierung des Fragebogens und des Inhalts des Fragebogens einfacher zu machen. Zusätzlich gehört dieser Fragebogen zu genau einem Projekt. Darüber hinaus enthält jeder Fragebogen eine Sammlung von beliebig vielen Fragebogen Versionen sowie Fragebogen Resultaten.

5.2.3 Fragebogenversion

Eine Fragebogenversion gehört zu genau einem Fragebogen. Diese Version enthält einen Identifikator um sie von anderen Versionen des zugehörigen Fragebogens zu unterscheiden. Der letzte Bestandteil der Fragebogenversion ist das Fragebogen Schema. Dies besteht aus einer JSON welche mit Hilfe des SurveyJS Frameworks im Browser zu einem funktionsfähigen Fragebogen gerendert werden kann.

5.2.4 Fragebogenresultat

Ein Fragebogenresultat gehört auch zu genau einem Fragebogen. Da ein Fragebogen verschieden Versionen haben kann, ist ein Verweis auf die entsprechende Fragebogenversion enthalten. Als letztes enthält ein Fragebogenresultat eine JSON, welche die Antworten einer Person auf diesen Fragebogen enthält.

5.3 Nutzer und Rollen

Um eine korrekte Authentifizierung bei der Interaktion zwischen Nutzer und System gewährleisten zu können. Werden verschiedene Arten von Nutzern benötigt.

Darüber hinaus werden innerhalb eines Projektes noch verschiedene Rollen für die Nutzer benötigt, um vor allem die in Abschnitt 4.2.3 genannte nicht funktionale Anforderungen des Datenschutzes und der Datenintegrität gewährleisten zu können.

5.3.1 Nutzer

Um eine einfache Benutzung des Systems zu ermöglichen, soll das System aus so wenig verschiedenen Nutzern wie möglich bestehen. Deshalb gibt es die drei Nutzergruppen: Gast, Nutzer und Administrator.

Gast Ein Gast ist ein nicht angemeldeter Nutzer oder ein Nutzer ohne Account. Er kann im Web-Client nur die Login- und Registerseiten aufrufen. Mit der Loginseite kann sich ein Gast mit bereits vorhandenem Nutzeraccount im System anmelden. Er wird nach erfolgreicher Anmeldung im System ein Nutzer. Mit der Registerseite kann sich ein Gast einen neuen Account im System erstellen. Dafür benötigt er allerdings einen Token, welcher nur von einem Administrator erstellt werden kann. Dies wird im System so umgesetzt, damit nur befugte Nutzer einen Account erstellen können. Nach erfolgreicher Registration wird der Gast im System eingeloggt und somit auch zum Nutzer.

Nutzer Ein Nutzer, welcher sich über den Web-Client im System angemeldet hat oder sich mit seinem Bearer-Token (entspricht API-Token) in der JSON-API authentifiziert hat, kann die meisten Funktionen des Systems nutzen. Diese kann er über die Schnittstellen des Web-Clients oder der JSON-API ausführen, solange er angemeldet ist.

Administrator Ein Administrator ist ein Nutzer mit zusätzlichen Administratorrechten. Er kann Register-Token für Gäste erstellen und den Gästen senden, damit diese sich einen neuen Nutzer Account erstellen können. Darüber hinaus kann er Nutzer sperren und löschen. Zusammenfassend ist ein Administrator ein Nutzer, welcher zusätzlich die Rechte hat bestimmte Nutzerdaten zu verwalten.

5.3.2 Rollen innerhalb eines Projektes

Mit Hilfe von diesen verschiedenen Arten von Nutzern können nun die Berechtigungen geprüft werden um die Funktionen innerhalb des Systems zu nutzen. Aller-

dings kann momentan jeder Nutzer auf alle Projektdaten im System zugreifen. Um zu verhindern das jeder Nutzer Zugriff auf alle Projekte hat, werden deshalb zusätzlich noch Rollen benötigt. Diese Rollen werden pro Nutzer und Projekt verteilt. Die Rechte des Nutzer Funktionen für Daten des Projektes zu nutzen, hängen dabei von der Rolle des Nutzers in diesem Projekt ab.

Jeder Nutzer der einem Projekt hinzugefügt wurde, kann dabei entweder die Rolle Projektverwalter oder Projektteilnehmer erhalten. Dabei sind Nutzer standardmäßig Projektteilnehmer und können von Projektverwaltern des Projektes zu Projektverwaltern werden. Die einzige Ausnahme hierfür stellt der Ersteller des Projektes dar. Dieser erhält standardmäßig die Rolle des Projektverwalters.

Projektteilnehmer Projektteilnehmer haben nur minimale Berechtigungen in dem jeweiligen Projekt. Sie besitzen meist nur Leserechte und können teilweise Ressourcen erstellen. Löschen und Editieren von Objekten kann jedoch nur der Projektverwalter.

Diese Rechte werden nur durch eine Ausnahme abgeändert. Der Ersteller eines Objektes kann diese ungeachtet der Rechte auch löschen oder editieren.

Projektverwalter Projektverwalter haben die Berechtigungen alle verfügbaren Funktionen des Systems innerhalb des Projektes zu nutzen. Sie können darüber hinaus Projektteilnehmer zu Projektverwaltern machen.

6 Aspekte der Implementierung

In diesem Kapitel werden zuerst in Abschnitt 6.1 die verwendeten und Technologien kurz erklärt und beschrieben. Danach folgt eine genaue Erklärung von zwei wichtigen Teilen der Implementierung.

6.1 Technologien

Die Arbeit wurde mit der Programmiersprache PHP vergleiche Abschnitt umgesetzt. Auf PHP aufbauend wurde das Framework Laravel genutzt, in Verbindung mit dem dort enthaltenen Laravel Passport, welche eine Implementierung der Autorisierung der API ermöglicht. Als Datenbank wurde für diese Arbeit MySQL genutzt. Die Umsetzung der Fragebögen und das Rendern dieser wird mit SurveyJS umgesetzt.

PHP PHP [11] ist eine serverseitige Programmiersprache. PHP ist die mit über 80% die am meisten genutzte Programmiersprache für Webseiten [12] (Stand September 2018).

Laravel Laravel [5] ist ein PHP Framework. Laravel bietet dabei aufbauend auf PHP ein gutes Grundgerüst um Webanwendungen zu entwickeln. Darüber hinaus wird es aktuell gehalten und beinhaltet auch neue Funktionen von PHP, wie Namespaces, Interfaces und Anonyme Funktionen [16].

Im Vergleich zu PHP wird das programmieren mit Laravel deutlich erleichtert. Laravel bietet viele Funktionen, die die Effizienz bei Entwickeln deutlich erhöhen. Einige Beispiele hierfür sind viele Helfer die für das Testen vorhanden sind. Mithilfe von Factories können Datenbanken zum Beispiel automatisch mit Daten zum testen gefüllt werden. Eine weitere nützliche Funktion ist das Routing in Laravel. In der Datei die für das Routing zuständig ist, kann man viele verschiedene Varianten nutzen um Routen zu erstellen. Man kann zum Beispiel standardmäßig die Route so

einrichten, dass eine Anfrage direkt an die entsprechende Methode im zugehörigen Controller weitergeleitet wird. Als weitere Variante können auch anonyme Funktionen genutzt werden. Das Routing kann auch genutzt werden um Middlewares dazwischen zu schalten. Diese können Funktionen wie zum Beispiel die Autorisierung übernehmen. Somit wird die Anfrage nur nach erfolgreicher Autorisierung weitergeleitet.

MySQL Datenbank MySQL [8] ist eine relationale Datenbank von Oracle. MySQL ist dabei robust, einfach zu nutzen und als freie Community Lizenz verfügbar [17].

Laravel Passport Laravel Passport [6] ist eine Open Source Implementierung des OAuth2 servers [10]. Es basiert auf dem OAuth2.0 Protokoll 3.2. Laravel Passport kann für eine einfache API Authentifizierung genutzt werden.

SurveyJS SurvyJS ist ein Javascript-Framework von Devsoft Balitc OU, welches genutzt werden kann um Fragebögen zu erstellen und im Browser rendern zu lassen. Es kann auch genutzt werden um auf diese Fragebögen zu Antworten. Bei der Implementierung des Projektes wurde SurveyJS als Basis für alle Fragebögen und Fragebogen-Schemas als Voraussetzung angenommen [14].

6.2 Zwei Aspekte der Implementierung

Zwei wichtige Aspekte der Implementierung sind der Web-Service und der Web-Client. Diese können beide von Nutzern genutzt werden um Ressourcen auf dem Server zu manipulieren. Das heißt sie können bei vorhandenen Berechtigungen die CRUD Funktionen auf Ressourcen ausführen. Web-Service und Web-Client besitzen fast komplett identische Funktionen.

Der Web-Service bietet eine Schnittstelle im System an mit der Nutzer-Clients oder externe Systeme mit dem System interagieren können. Dabei werden dem Entwurf und der Implementierung des Nutzer-Clients fast keine Vorgaben gemacht. Ersteller eines Nutzer-Clients können diesen daher genau an ihre Bedürfnisse anpassen. Lediglich das Format der Anfragen an die Schnittstelle und Antworten von der Schnittstelle müssen die vom System definierte Form besitzen (vgl. Abschnitt...) damit die Nutzer-Clients erfolgreich mit dem System interagieren können.

Der Web-Client dagegen ist eine Implementierung für einen webbasierten Nutzer-Client. Die Nutzer des Systems sind deshalb nicht darauf Angewiesen einen eigenen Client zu entwickeln um das System nutzen zu können. Die Nutzung des Web-Clients ist dafür auch nicht an alle Bedürfnisse der Nutzer angepasst.

6.2.1 Implementation Web-Service

Ein wichtiger Teil der Implementierung ist der Web-Service des Systems. Dieser wurde bei dieser Arbeit durch eine API umgesetzt. Die API basiert dabei auf der REST Architektur (vgl. Abschnitt 3.3). Das heißt sie ist zustandslos und speichert deshalb keine Daten zwischen Anfragen (entspricht Requests durch Client an das System) ab. Um diese Zustandslosigkeit zu erreichen, muss deshalb jede Anfrage alle Daten enthalten, welche benötigt werden um die gewünschte Funktion des Systems auszuführen. Die Antwort (entspricht Response des Systems an den Client) des Systems auf diese Anfrage muss wiederum alle nötigen Informationen beinhalten, damit der Client, welcher die API nutzt alle erforderlichen Informationen erhält um den Erfolg und die Daten zu analysieren (vgl. Abschnitt).

Die Daten, welche bei Anfragen und Antworten zwischen Nutzer-Client und System übertragen werden, liegen dabei im JSON-Format vor. Das JSON-Format basiert auf Name-Wert Paaren. Dabei beschreibt der Name die Zugehörigkeit des jeweils entsprechenden Wertes.

Routen

In Laravel werden Routen genutzt, um HTTP Anfragen an URIs zu entsprechenden Funktionen weiterzuleiten. Die Routen für APIs befinden sich standardmäßig in der Datei api.php.

Das Listing 6.1 enthält einen Ausschnitt aus dieser Datei. Die Route in Zeile zwei des Listings 6.1 ist dabei eine Standardvariante einer Route. In dieser wird ein HTTP-GET Anfrage über den Link /projects an die Index-Methode im ProjectController weitergeleitet. Die Route in Zeile zwei und drei ist eine Ressource-Route. Ressource-Routen sind eine Möglichkeit wie man in Laravel, alle CRUD-Methoden auf einer Ressource gruppiert, in einer Route zusammenfassen kann. In der Route wird zusätzlich noch die Middleware 'auth' eingebunden. Diese Middleware überprüft bei allen auf dieser Route eingehenden Anfragen, ob der ausführende Nutzer angemeldet ist und somit die Berechtigung besitzt, diese Route auszuführen.

```
'headers' = {
Route::get('projects', 'ProjectController@index');
Route::middleware('auth')->
        resource('surveys','SurveyController');
}
```

Listing 6.1: Routing

```
'headers' = {
        'Accept' = 'application/json',
        'Content-Type' = 'application/json',
        'Authorization' = 'Bearer '.$accessToken,
}
```

Listing 6.2: JSON-Request

Anfrage

Anfragen an das System werden von Nutzern oder externen Systemen ausgeführt. Dabei werden über die Routen Daten von den Nutzern an das System gesendet. Dies führt darauf, wenn möglich, die vom Nutzer durch die gewählte Route spezifizierte Funktion im System aus.

Erstellen einer Anfrage

Die Daten die in einer Anfrage vom Nutzer an das System übermittelt werden, müssen im JSON-Format gesendet werden. Die Nutzer der Schnittstelle müssen deshalb im Header ihrer HTTP-Anfrage an das System die Felder 'Accept' und 'Content-Type' mit den entsprechenden Werten aus Listing 6.2 gesetzt werden.

Damit ein Nutzer erfolgreich auf das System zugreifen kann, muss sich dieser mit seinem API-Token authentifizieren. Das System überprüft diesen Token mit Hilfe von OAuth2. Damit das System den API-Token des Nutzers erkennen und überprüfen kann, muss im Header der HTTP-Anfrage ein Feld mit dem Name Authorization und Wert 'Bearer '. gefolgt von dem API-Token des Nutzers gesetzt werden. Die API Middleware, welche die API-Authorisierung mit OAuth2 enthält, überprüft bei allen Anfragen über die API-Routen ob die Anfrage von einem Nutzer des Systems erstellt wurde.

```
1 {
2         $projects = App\Projects::all();
3         $response_code = '200';
4         return response()->json(
5                 'response-code => $response_code,
6                 'projects' => $projects);
7 }
```

Listing 6.3: Abrufen aller Projekte

Verarbeiten einer Anfrage

Nachdem ein Nutzer über die Schnittstelle an das System eine Anfrage geschickt hat, wird diese nach erfolgreicher Authentifizierung an die entsprechende Funktion im zur Ressource gehörenden Controller weitergeleitet.

Um zu überprüfen, ob der Inhalt der Anfrage das JSON-Format besitzt, wird als erstes in der Funktion überprüft, ob der Content-Type im Header den Wert 'application/json' enthält.

Ist dies der Fall, so findet die entsprechende Verarbeitung der Anfrage in der Funktion statt. Nachfolgend finden sich fünf Beispiele für Funktionen mit Erklärung, welche vom System über die API ausgeführt werden können. Die Codebeispiele sind dabei Ausschnitte aus dem fertigen System. In diesem ist unter anderem die Überprüfung der Rechte der Nutzer nicht enthalten, der Fokus liegt auf der Implementierung der Verarbeitung der Ressourcen. Zuerst wird die Implementierung des Abrufens aller Projekte eines Nutzer erklärt. Daraufhin befinden sich in den folgenden Abschnitten je ein Beispiel mit Erklärung für jede CRUD-Funktion.

Abrufen aller Projekte Listing 6.3 enthält ein Teil des Codes der zum Abrufen aller Projekte eines Nutzers nötig ist. In Zeile zwei werden mit Laravel Eloquent alle gespeicherten Projekte abgerufen. Wenn bis dahin kein Fehler aufgetreten ist, wird der Statuscode der mit der Antwort der API zurückgegeben wird auf alles ok gesetzt und zusammen mit den Abgerufenen Projekten abgeschickt.

Abrufen eines Fragebogens Listing 6.4 enthält ein Teil des Codes der zum Abrufen eines Fragebogens nötig ist. Anfangs wird in Zeile zwei die ID des Fragebogens aus dem, im HTML-Formular übergebenen Werten, herausgefiltert. In Zeile drei

```
{
        $survey_id = $request->input('survey_id');
        $survey = App\Suvery::find($survey_id);
        $response_code = '200';
        return response()->json(
                'response-code => $response_code,
                'survey' => $survey);
}
```

Listing 6.4: Abrufen eines Fragebogens

```
{
$survey_id = $request->input('survey_id');
$version = $request->input('version');
$survey_schema = $request->input('survey_schema');
$survey_version = [
        'version' => $version,
        'data' => $data,
        'survey_id' = $survey_id,
        'creator_id' = Auth::id();
]
$survey = App\Suvery::create($survey_version);
$response_code = '201';
return response()->json(
        'response-code => $response_code);
}
```

Listing 6.5: Erstellen einer Fragebogenversion

wird daraufhin der Fragebogen mit der übergebenen ID aus der Datenbank geladen. Abschließend werden auch hier wieder der Statuscode gesetzt und die JSON Antwort wird mit Statuscode und der geladenen Fragebogen zurückgegeben.

Erstellen einer Fragebogenversion Um einen Fragebogen zu erstellen, filtert man zuerst, wie in Listing 6.5 die einzelnen Werte der zu erstellenden Ressource heraus. Danach erstellt man eine Ressource mit diesen Werten. Abschließend erstellt man dann diese wie in Zeile 11 in der Datenbank.

```
1  {
2  $id = $request->input('surver_version_id');
3  $survey_id = $request->input('survey_id');
4  $version = $request->input('version');
5  $survey_schema = $request->input('survey_schema');
6  $survey_version->version => $version;
7  $survey_version->schema => $survey_schema;
8  $survey_version->survey_id  => $survey_id;
9  $survey_version->save();
10 $response_code = '200';
11 return response()->json(
12         'response-code' => $response_code,
13         'survey_version' => $surver_version);
14 }
```

Listing 6.6: Editieren einer Fragebogenversion

```
1  {
2          $id = $request->input('survey_id');
3          $projects = App\Projects::find($survey_id);
4          $projects->delete();
5          $response_code = '201';
6          return response()->json(
7                  'response-code => $response_code);
8  }
```

Listing 6.7: Löschen eines Fragebogens

Editieren einer Fragebogenversion Listing 6.6 enthält einen Teil des Codes zum ändern einer Ressource. Man filtert zuerst die Werte heraus, welche die Ressource annehmen soll. Dann lädt man die Ressource und ändert die Werte derer auf die neuen Werte. Abschließend speichert man die geänderten Werte wie in Zeile neun ab.

Löschen eines Fragebogens Zum löschen einer Ressource lädt man zuerst wie in Zeile zwei des Listings 6.7 die Ressource aus der Datenbank. Dann führt man die delete Methode aus Zeile vier auf dieser Ressource aus.

```
1  {
2  public function validateProject(Request $request) {
3          $this->validate($request, [
4                  'title' =>
5                          'required|string|
6                          unique:projects,title|max:255',
7              'description' =>
8                          'nullable|string',
9                  ]);
10         }
11
12 public function store(Request $request) {
13         $this->validateProject($request);
14 }
15 }
```

Listing 6.8: Validierung der Projektressource

Validieren von Anfragen Laravel bietet mehrere Möglichkeiten um eine Validierung von Ressourcen umzusetzen. In der Implementierung dieser Arbeit wurde eine Methode im entsprechenden Controller erstellt, welche alle Attribute, welche über einfache Eingaben gesetzt werden können, zu validieren. In diesem Fall wird überprüft, ob der Titel des Projektes vorhanden ist, sein string ist, einmalig in der Spalte title in der Tabelle projects ist und maximal 255 Zeichen lang ist. Die Beschreibung darf den Wert Null annehmen und muss ein string sein.

In Zeile 13 wird dann die Validierungsmethode auf die eingehende Anfrage angewandt. Im Falle von Fehlern wird die Ausführung der Methode automatisch abgebrochen und die Fehler werden in der Antwort zurückgegeben.

Antwort

Eine Antwort auf eine entsprechende Anfrage wird von der Funktion im Controller nach der Verarbeitung an den Nutzer zurückgegeben. Die Antwort des Systems besteht aus einer HTTP-Antwort welche an den Nutzer gesendet wird. Der Inhalt der Antwort besteht wie auch der Inhalt der Anfrage aus Daten im JSON-Format. Sie enthält immer das Feld mit dem Namen 'success' mit einem Statuscode als Wert (entspricht Response-Code in Abbildung 6.9, der Angibt ob die Verarbeitung der Anfrage im System erfolgreich war. Eine Auflistung der wichtigsten Statuscodes

```
{
        'Response-Code' = 200,
        'project' = [
                'id' = 2,
                'title' = 'Testprojekt',
                'description' = 'Kurze Beschreibung.',
                'survey_ids' = [
                        2,
                        3,
                        4,
                ]
        ]
}
```

Listing 6.9: JSON-Response

befindet sich im Nachfolgenden Abschnitt . Bei erfolgreichen Verarbeitungen der Anfrage, kann es sein das zusätzliche Daten in der JSON enthalten sind. Dies kann zum Beispiel die Ressource sein, wenn die Anfrage diese vom System abrufen wollte.

Abbildung 6.9 ist ein Beispiel, für eine Antwort auf eine erfolgreiche Ausführung der Anfrage. Der Nutzer hat hierbei eine Anfrage an das System gestellt, bei der er das Projekt mit der ID 2 vom System abrufen wollte. Da die Anfrage korrekt war und der Nutzer die Rechte hatte dieses Projekt abzurufen, konnte die Anfrage erfolgreich ausgeführt werden. Deshalb hat die Antwort als Inhalt zum einen den Statuscode 200 (Anfrage hat funktioniert) und zum anderen die Daten des Projektes mit der ID 2.

Statuscode Die Statuscodes die das System in Antworten auf Anfragen zurücksendet basieren auf den HTTP-Statuscodes [4]. Sie sind in der Antwort im JSON-Feld mit dem Name 'Response-Code' als Wert enthalten. In Tabelle 6.1 befindet sich ein Überblick über die wichtigsten Statuscodes, welche in Antworten des Systems genutzt werden.

6.2.2 Implementation Web-Client

Um den Nutzern eine Möglichkeit zu geben mit dem System ohne eine API zu interagieren wurde im Verlauf der Arbeit ein Web-Client erstellt, welche eine Ver-

Code	Verwendung
200	Request hat funktioniert.
201	Request war erfolgreich und die Ressource wurde erstellt.
204	Request hat funktioniert, Server muss keinen Inhalt zurückgeben.
400	Request wurde nicht verstanden.
401	Request benötigt Nutzer Autorisierung.
403	Request ist nicht erlaubt.
404	Request URI wurde nicht gefunden.

Tabelle 6.1: Antwort Statuscodes (basierend auf [4])

wendung der wichtigsten Funktionen der API auch über HTML-Seiten im Browser ermöglicht.

In den folgenden Abschnitten wird genauer auf die Implementation des Web-Clients eingegangen anhand des Beispiels erstellen und anzeigen einer neuen Fragebogenversion eines neuen Fragebogens. Bevor der Nutzer dies ausführen kann, muss er sich zuerst im System anmelden.

Dashboard Das Dashboard ist die Seite auf die ein Nutzer automatisch nach erfolgreicher Anmeldung über den Web-Client weitergeleitet wird. Auf dieser Seite sind alle Projekte des Nutzers mit Titel und kurzer Beschreibung aufgelistet. Der Nutzer kann auf die einzelnen Projekte klicken, um auf die jeweilige Seite des Projektes zu gelangen.

Projekt Auf dieser HTML-Seite befinden sich der Titel und die Beschreibung des Projektes, sowie eine Liste aller zugehörigen Fragebögen mit Titel und Beschreibung. Auf die einzelnen Fragebögen kann der Nutzer klicken um auf die Seite des Fragebogens weitergeleitet zu werden.

Wenn der Nutzer die Rolle eines Projektleiters für dieses Projekt besitzt, hat er zusätzlich die Möglichkeit das Projekt zu verwalten.

Fragebogen Auf dieser HTML-Seite befinden sich der Titel und die Beschreibung des Fragebogens, sowie eine Liste aller Fragebogenversionen. Der Nutzer kann auf die einzelnen Fragebogenversionen klicken, um auf die jeweiligen Seiten von diesen weitergeleitet zu werden. Zusätzlich befindet sich auf dieser HTML-Seite ein Link den der Nutzer klicken kann um eine neue Version des Fragebogens zu erstellen.

Ist der Nutzer Projektleiter des Projektes, zu dem der Fragebogen gehört, hat dieser zusätzlich die Möglichkeit den Fragebogen zu verwalten und mit anderen Nutzern zu teilen. Darüber hinaus hat er die Möglichkeit eine Sammlung aller Fragebogenresultate für diesen Fragebogen abzurufen.

Fragebogenversion erstellen Diese HTML-Seite besteht aus einem HTML-Formular mit dem man die JSON eines SurveyJS-Fragebogens auf der Plattform abspeichern kann. Dabei findet eine clientseitige Validierung der JSON statt, welche überprüft ob aus der JSON ein gültiger SurveyJS-Fragebogen erstellt werden kann.

Fragebogenversion abrufen Nachdem die neue Fragebogenversion erfolgreich auf dem Server gespeichert wurde, wird der Nutzer auf die HTML-Seite der erstellten Fragebogenversion weitergeleitet. Dabei wird die Fragebogenversion direkt mit Hilfe von SurveyJS gerendert dargestellt.

7 Anforderungsabgleich

Hier werden die in Kapitel 4 aufgestellten Anforderungen bewertet. Anhand der folgenden Skala wird überprüft, ob die Anforderungen implementiert oder ob sie nur konzipiert wurden. Darüber hinaus wird bei der Implementierung unterschieden, ob die Anforderungen komplett oder nur teilweise implementiert wurden.

7.1 Funktionale Anforderungen

In diesem Abschnitt werden die funktionalen Anforderungen, welche in Abschnitt 4.2.2 aufgestellt werden auf Vollständigkeit der Erfüllung im fertigen System überprüft. Für die Analyse wurde die nachfolgende Skala benutzt. Bei erfüllten Anforderungen wird unterschieden ob die Anforderungen vollständig oder nur teilweise erfüllt wurden. Wenn die Anforderungen nicht erfüllt werden, wird unterschieden ob diese konzipiert wurden, jedoch nicht erfüllt, oder ob sie gar nicht erst bearbeitet wurden.

++	Anforderung wurden vollständig erfüllt
+	Anforderung wurde teilweise erfüllt
0	Anforderung wurde konzipiert
-	Anforderung wurde nicht erfüllt

Nr.	Nicht funktionale Anforderung	Erfüllungsgrad
Gast:		
FA1.01	Gast kann sich registrieren	++
FA1.02	Gast kann sich anmelden	++
Nutzer:		
FA1.03	Nutzer kann API-Key erstellen	++
FA1.04	Nutzer kann API-Key erneuern	++
FA1.05	Nutzer kann API-Key löschen	++
FA1.06	Nutzer kann neues Projekt erstellen	++
Administrator:		
FA1.07	Administrator kann neuem Nutzer Register-Token schicken	++
FA1.08	Administrator kann Nutzer sperren	++
FA1.09	Administrator kann Nutzer löschen	++

Tabelle 7.1: Erfüllungsgrad nutzerabhängige funktionale Anforderungen

Nr.	Nicht funktionale Anforderung	Erfüllungsgrad
Projektverwalter:		
FA2.01	Projektverwalter kann Projekte bearbeiten	++
FA2.02	Projektverwalter kann Projekte löschen	++
FA2.03	Projektverwalter kann Nutzer als Projektteilnehmer zu Projekt hinzufügen	++
FA2.04	Projektverwalter kann Projektteilnehmer zu Projektverwalter des Projektes machen	++
FA2.05	Projektverwalter kann CRUD Fragebogen innerhalb des Projektes	++
FA2.06	Projektverwalter kann CRUD Fragebogen-Version innerhalb einem Fragebogen des Projektes	++
FA2.07	Projektverwalter kann Fragebogen-Resultat zu einem Fragebogen des Projektes hinzufügen	++
FA2.08	Projektverwalter kann Fragebogen-Resultat-Kollektion zu einem Fragebogen des Projektes abrufen	++
FA2.09	Projektverwalter kann Fragebogen-Schema mit anderen Nutzern teilen	++
Projektteilnehmer:		
FA2.10	Projektteilnehmer kann Projekte abrufen	++
FA2.11	Projektteilnehmer kann Fragebögen des Projektes abrufen	++
FA2.12	Projektteilnehmer kann Versionen eines Fragebogens des Projektes abrufen	++
FA2.13	Projektteilnehmer kann Version eines Fragebogens des Projektes erstellen	++
FA2.14	Projektteilnehmer kann Resultat eines Fragebogens des Projektes erstellen	++

Tabelle 7.2: Erfüllungsgrad projektspezifische funktionale Anforderungen

Nr.	Nicht funktionale Anforderung	Erfüllungsgrad
Fragebögen:		
FA3.01	Fragebogen hat Versionsstände	++
FA3.02	Fragebogen wird validiert	++
FA3.03	Fragebogen Schema kann über HTML abgerufen werden	++
FA3.04	Fragebogen Schema kann als JSON abgeurfen werden	++

Tabelle 7.3: Erfüllungsgrad allgemeine funktionale Anforderungen

7.2 Nicht funktionale Anforderungen

In diesem Abschnitt werden die nicht funktionalen Anforderungen die in Abschnitt 4.2.3 aufgestellt wurden auf Vollständigkeit der Erfüllung der Anforderungen überprüft. Für die Analyse der Erfüllung wird dieselbe Skala wie in Abschnitt 7.1 benutzt.

++	Anforderung wurden vollständig erfüllt
+	Anforderung wurde teilweise erfüllt
0	Anforderung wurde konzipiert
-	Anforderung wurde nicht erfüllt

Nr.	Nicht funktionale Anforderung	Erfüllungsgrad
NFA1	Datenschutz	++
NFA2	Datenminimierung	++
NFA3	Verfügbarkeit	++
NFA4	Datenintegrität	++
NFA5	Skalierbarkeit	+
NFA6	Robustheit	++

Tabelle 7.4: Erfüllungsgrad nicht funktionale Anforderungen

8 Zusammenfassung und Ausblick

Dieses Kapitel enthält eine abschließende Zusammenfassung der Arbeit. In Abschnitt 8.2 folgt ein kurzer Ausblick auf zukünftige Anwendungen und mögliche Erweiterungen.

8.1 Zusammenfassung

Im Verlauf dieser Arbeit wurde eine zentrale Plattform zum Speichern und Verwalten von Fragebögen konzipiert und realisiert.

Der Anfang der Arbeit besteht aus Kapitel 1 bis Kapitel 3, welche eine Einführung in die Arbeit darstellen. Dabei wird in Kapitel 1 die Arbeit motiviert und die zentrale Problemstellung beschrieben. Kapitel 2 und 3 behandeln verwandte Arbeiten und die Grundlagen der Arbeit. Daraufhin folgt die Planung der Arbeit in den Kapiteln 4 und 5. Im Kapitel 4 werden die Anforderungen an das System aufgestellt. Das Kapitel 5 enthält den Entwurf des Systems. Danach folgt mit Kapitel 6 die Implementierung des zuvor geplanten Systems. Abschließend folgt in Kapitel 7 eine Überprüfung der Umsetzung der in Kapitel 4 aufgestellten Anforderungen.

Die im Laufe der Arbeit erstellte Plattform bietet dabei die Möglichkeit, Projekte anzulegen und zu bearbeiten. Innerhalb von diesen Projekten können Fragebögen mit verschiedenen Versionen und Antworten abgespeichert werden. Zudem können weiter Nutzer zu diesen Projekten hinzugefügt werden.

Dies bietet den Nutzern des Systems die Möglichkeit, Fragebögen in Projekte zu gruppieren, zu speichern und zu verwalten. Diese Fragebögen können auch mit anderen Nutzern des Systems geteilt werden. Jeder Nutzer welcher einem Projekt zugeordnet ist, kann diese Inhalte dieses je nach Berechtigung verwalten.

Da diese Arbeit speziell zur Forschung gerichtet motiviert war Somit kann diese Plattform genutzt werden um Forschungsergebnisse zentrale zu speichern und mit anderen Forschern zu teilen. Für die Nutzer des Systems besteht darüber hinaus

die Möglichkeit andere Forscher zu ihren Projekten hinzuzufügen. So können Forschungsergebnisse einfach geteilt oder auch gemeinsam erstellt und bearbeitet werden.

8.2 Ausblick

Die im Verlauf der Arbeit erstellte Plattform ist webbasiert und bietet eine RESTful-API als primäre Schnittstelle an. Somit ermöglicht dies Entwicklern einfach Erweiterungen der Plattform wie zum Beispiel eine mobile Anwendung für Smartphones zu entwickeln. Darüber hinaus kann diese Plattform von bereits vorhandenen fragebogenbasierten Forschungsprojekten als zentrale Plattform für die Verwaltung von den Fragebögen genutzt werden.

Literaturverzeichnis

[1] *Abbildung MobileTx*. https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.030/research/MobileTX/architecture.jpg, . – Zuletzt abgerufen am: 06.09.2018

[2] *Abbildung QuestionSys*. https://www.uni-ulm.de/fileadmin/_processed_/9/f/csm_architecture_193163fb91.png, . – Zuletzt abgerufen am: 06.09.2018

[3] *CRUD*. https://en.wikipedia.org/wiki/Create,_read,_update_and_delete, . – Zuletzt abgerufen am: 27.08.2018

[4] *HTTP Statuscodes*. https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html, . – Zuletzt abgerufen am 27.08.2018

[5] *Laravel*. https://laravel.com/, . – Zuletzt abgerufen am: 24.08.2018

[6] *Laravel Passport*. https://laravel.com/docs/5.6/passport, . – Zuletzt abgerufen am: 07.09.2018

[7] *MobileTx*. https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/mobiletx/, . – Zuletzt abgerufen am: 06.09.2018

[8] *MySQL*. https://www.mysql.com/, . – Zuletzt abgerufen am: 27.08.2018

[9] *OAuth*. https://oauth.net/, . – Zuletzt abgerufen am: 14.08.2018

[10] *OAuth Github*. https://github.com/thephpleague/oauth2-server, . – Zuletzt abgerufen am: 14.08.2018

[11] *PHP*. https://secure.php.net/, . – Zuletzt abgerufen am: 05.09.2018

[12] *PHPdetails*. https://w3techs.com/technologies/details/pl-php/all/all, . – Zuletzt abgerufen am: 07.09.2018

[13] *QuestionSys*. https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/questionsys/, . – Zuletzt abgerufen am: 06.09.2018

[14] *SurveyJS*. https://surveyjs.io/, . – Zuletzt abgerufen am: 27.08.2018

[15] *TrackYourTinnitus*. `https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/trackyourtinnitus/`, . – Zuletzt abgerufen am: 06.09.2018

[16] BEAN, M. : *Laravel 5 Essentials*. Packt Publishing, 2015. – ISBN 9781785283017

[17] BELL, C. : *MySQL for the Internet of Things*. 1st. Berkely, CA, USA : Apress, 2016. – ISBN 1484212940, 9781484212943

[18] BOYD, R. : *Getting Started with OAuth 2.0*. O'Reilly Media, Incorporated, 2012 (Oreilly and Associate Series). – ISBN 9781449311605

[19] FIELDING, R. T.: *Architectural Styles and the Design of Network-based Software Architectures*, Diss., 2000. – AAI9980887

[20] PRYSS, R. ; SCHLEE, W. ; LANGGUTH, B. ; REICHERT, M. : Mobile Crowdsensing Services for Tinnitus Assessment and Patient Feedback. In: *6th IEEE International Conference on AI & Mobile Services (IEEE AIMS 2017)*, IEEE Computer Society Press

[21] SCHICKLER, M. ; PRYSS, R. ; SCHOBEL, J. ; REICHERT, M. : Supporting Remote Therapeutic Interventions with Mobile Processes. In: *6th IEEE International Conference on AI & Mobile Services (IEEE AIMS 2017)*, IEEE Computer Society Press

[22] SCHICKLER, M. ; PRYSS, R. ; STACH, M. ; SCHOBEL, J. ; SCHLEE, W. ; PROBST, T. ; LANGGUTH, B. ; REICHERT, M. : An IT Platform Enabling Remote Therapeutic Interventions. In: *30th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2017)*, IEEE Computer Society Press

[23] SCHOBEL, J. ; PRYSS, R. ; SCHICKLER, M. ; RUF-LEUSCHNER, M. ; ELBERT, T. ; REICHERT, M. : End-User Programming of Mobile Services: Empowering Domain Experts to Implement Mobile Data Collection Applications. In: *5th IEEE International Conference on Mobile Services (MS 2016)*, IEEE Computer Society Press, 1–8

Name: Fabian Egl Matrikelnummer: 866311

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Fabian Egl